EECE.3220 Spring 2019: Exam 3
Class Definitions and ADT Descriptions

Class definition for Section 1

```
typedef double QueueElement;

class Queue {
public:
   Queue(unsigned maxSize = 1024);     // Constructor
   ~Queue();                           // Destructor
   bool empty() const;                 // Returns true if queue empty
   void enqueue(const QueueElement &val);   // Add val to back of queue
   void dequeue();                          // Remove head of queue
   QueueElement getFront();            // Read value at front of queue
   unsigned numVals();                 // Return # values in queue
                                       //  NEW FUNCTION FOR EXAM 3
   void display(ostream &out);         // Print values in entire queue
                                       //  NEW FUNCTION FOR EXAM 3
private:
   QueueElement *list;// The actual data stored in the queue
   int front, back;   // Indexes for head & tail of queue
   unsigned cap;      // Capacity (max size) of queue
}
```

Class definition for Section 1

```
typedef double QueueElement;

class Queue {
public:
   Queue();                            // Constructor
   ~Queue();                           // Destructor
   bool empty() const;                 // Returns true if queue empty
   void enqueue(const QueueElement &val);   // Add val to back of queue
   void dequeue();                     // Remove head of queue
   QueueElement getFront();            // Read value at front of queue
   bool isPriorityQ();                 // Return true if queue sorted
                                       //  from high to low value
                                       //  NEW FUNCTION FOR EXAM 3
private:
   class Node {            // Queue node
   public:
      QueueElement data;
      Node *next;
   };
   Node *front, *back;     // Addresses of front/back of queue
};
```

Class definition for Section 3

```
class LList {
public:
   LList();                        // Default constructor
   ~LList();                       // Destructor
   bool isEmpty();                 // Returns true if list is empty
   void display(ostream &out);     // Print LList contents
   void insert(double v);          // Add new value to list
   void remove(double v);          // Remove node with matching value
   bool LList::inList(double val);     // Return true if val in list
                                       //  NEW FUNCTION FOR EXAM 3
private:
   class Node {
   public:
      double val;      // Value in each node
      Node *next;      // Pointer to next node
   };

   Node *first;  // Pointer to first node
};
```